Reasonably Optimal Utilisation Through Evolution (ROUTE) in Airspace Design

Thomas Lawson[1] and Yanyan Yang[2]

[1,2] School of Engineering, University of Portsmouth, Portsmouth, UK
[1]up780962@myport.ac.uk [2]linda.yang@port.ac.uk

Abstract: The underlying navigation technology that enables the navigation of aircraft through airspace is improving, allowing aircraft to navigate waypoints that do not need to be previously defined, no longer being confined to antiquated navigational aids. From this the opportunity is arising to create highly optimized airspace that can change its design on the fly in reaction to the environment. This paper presents a novel approach towards route finding in airspace design using computational evolution. The proposed method, ROUTE, can create new airspace designs in reaction to changing environmental constraints, optimising for fuel burn (therefore cost and emissions) as well as noise disturbance, and route length.

Keywords: Airspace Design, Genetic Algorithms, Dynamic Airspace, Constrained Optimisation.

1 Introduction

Currently, global airspace infrastructure is built around 'conventional' navigation, making use of ground-based beacons. Airspace design is carried out by humans who are aided by computer design tools. As there are many factors that must be considered such as noise pollution, airport capacity and aircraft fuel consumption which impacts fuel cost and emissions of the aircraft. All these factors are compounded by an ever-increasing number of aircraft. This conventional approach does not take advantage of modern navigation technologies [1], [2]. With the advent of modern navigation technologies, aircraft far better know where they are, this means that a flight is no longer reliant on flying passed fixed waypoints. This development and introduction of Performance-based Navigation gives aircraft the opportunity to fly more dynamic routes that can change and react to environmental constraints, for example weather.

Researchers have proposed methods for using genetic algorithms for the navigation of Unmanned Aerial Vehicles (UAVs) [3], [4]; work has also been done to look into airspace sectorisation using recursive geometric optimisation [5] as well as evolutionary optimisation [6]. The UKs Civil Aviation Authority (CAA) have outlined their view of how modern navigation should be used to better use airspace [7].

This paper proposes an evolutionary heuristic computer aided method, called ROUTE, for developing airspace. Genetic algorithms are computational models for evolution designed to mimic natural evolution, where a pool of candidates is populated

with randomly created individuals. A pool of breeding candidates from each generation are selected using a roulette wheel selection method. This ensures preference for more fit candidates, however means that less fit candidates can also be selected. They are then mutated and crossed over and create the next generation. This method of selecting more fit individuals, while maintaining diversity from the fewer less fit individuals, means with each generation the algorithm will produce individuals whose fitness will tend towards the optimal solution [8].

A benefit of using a computational approach means ROUTE can work alongside performance based [7] navigation to create automated reactive airspace, not dependant on constraints such as fixed waypoints. New waypoints can be transmitted to aircraft en-route. It would reduce the human resource needed for airspace design, as well as allowing numerous contingent situations to be more readily modelled.

The rest of the paper is organised as follows. Related work is critically reviewed in Section 2. Section 3 introduce the design of the algorithm. Section 4 describes the algorithm in detail. In Section 5 we present the experimental results and discuss issues raised from them. Section 6 concludes the paper and propose future work.

2 Related Work

Work has been done to use GAs to compute the routes for UAVs [3], [4]. This work uses real time data to build the route for one UAV in less than half a second. While this work looks at the problem of computing a route for an aircraft based on terrain and avoiding certain features, it is limited in as much as it does not look at repeatable routes, or how one route may interact with another, other than coordinating the arrival between two drones.

Analysis has been done [9], and a method proposed for organising airspace into 'tubes' this approach means reducing complexity as aircraft will fly through tubes in the sky. The tubes will be formed using the Hough Transform, then using a GA adjusted to fit in the highest percentage of flights, with minimum extra distance travelled [10]. This approach looks at how to capture large routes aircraft can be moved through, however it doesn't look at how aircraft should be routed from en-route airspace to an airfield. While this may not be such a problem in more sparsely populated countries, such as the US, this is more of a concern in the UK as airports are often in densely populated areas.

Researchers also look at how airspace can be more effectively sectorised [5] and how this can be done dynamically [11]. The aforementioned research looks at how to reduce complexity and optimize airspace by changing the shape of sectors, but it does not consider the benefits that can be gained from changing the underlying routes aircraft fly.

There has also been research into how airspace can be more efficiently utilised through Area Navigation, a technique which allows for dynamic navigation points. This research shows that more flexible navigation can allow for more efficient routes [12]. This paper proposes a method to determine the most appropriate points evolutionarily.

A technique of optimising approach trajectories based on environmental constraints has also been proposed [13]. This is improved upon [14]. This research looks at

methods for proposing routes which build on existing routes. Our method proposes an approach capable of proposing its own routes.

Lastly it has been shown that through optimising speed and trajectories it is possible to improve efficiency and reduce emissions of a flight path [15], [16]. Our evolutionary approach hopes to iteratively find these improvements and propose routes that apply them.

The proposed ROUTE algorithm offers a technique building on past research to build routes which form the best compromise between Speed, efficiency, pollution, and therefore cost. The goal of the ROUTE algorithm is to address these problems and develop a technique for generating airspace designs in a reasonable amount of time, to be able to better take advantage of more modern navigation technologies and adapt in real time to changes in the environment.

3 The ROUTE Design

3.1 Design Objectives

The ROUTE algorithm aims to find the optimal design of airspace, consisting of multiple routes, given a set of defined environmental constraints. It will look to use computational evolution to propose the most effective airspace structure given a set of constraints. This can be applied on a smaller scale to produce Arrival and Departure routes or a larger scale to produce the design for a country's airspace or beyond. The design objectives have been outlined as follows:

- Below 4,000ft priority is given to minimising noise pollution over populated areas.
    - As an aircraft is not at this altitude for long, and this is the altitude that most noise pollution is produced.
- Between 4,000ft and 7,000ft equal priority is given to minimising noise pollution and route efficiency.
    - At this altitude it is still important to consider surrounding communities, however if this might produce an unnecessarily complex route, a compromise would be preferred.
- Above 7,000ft priority is increasingly given to maximising route efficiency.
    - As an aircraft climbs away from 7,000ft its noise pollution becomes less of a concern, and for this reason simpler more efficient routes are preferred.
    - Route efficiency can be described as minimising journey times and route lengths as well as fuel burn by aircraft.
- Maximise airport utilisation
    - Where capacity is sufficiently large, aircraft should arrive at an airport at the maximum rate safely allowed.
- Minimise Airspace complexity

    - Many separate routes, which cross each other produce risk for air traffic control and pilots, for this reason flightpaths should cross each other as infrequently as possible.
- It must be possible to avoid designated areas.
    - Due to weather or security concerns, for example, or regulatory issues such as overflying national parks.

Maximising efficiency and airport utilisation should result in the ROUTE algorithm finding novel approaches to managing arrivals, without placing aircraft on hold as this is a highly inefficient. An example of how improved efficiency can be achieved is seen in LAMP 1a, which uses an arc to organise arrivals into London City airport [17]. Although only on one route, and done manually, this demonstrates that there are approaches beyond using holding stacks to improve efficiency. The proposed method ROUTE will propose ways that these optimisations could be automated, and also find many small improvements which can accumulate.

3.2 Genome Structure

Each generation the algorithm will produce a population of multiple possible solutions, it is these solutions which will be scored based on its fitness and be acted upon by the genetic operators. Each solution contains a set of n routes, where n is the number of pairs of entrance and exit (start and finish) points. Each solution will contain the same number of routes between the same points, only one route within a solution will be subject to mutation or crossover.

3.3 Mutators

The ROUTE algorithm involves 6 genetic operators, one crossover operator, and 5 genetic mutators. These mutators are used to introduce suitably variance to allow routes to converge towards an optimal solution. All operators are only applied to the intermediary points on a route, the first and final points remain static. For each solution, only one route will have the chance of being mutated. Now we will outline the six operators.

1. *Crossover*: This operator combines two parent routes into two new offspring. A random point is selected along each route, the first half of the first route is then combined with the first part of the second route, and the second part of the first route is combined with the second part of the second route. This operator is only applied to routes between the same start and finish point. The number of nodes in the child routes can differ from their parents.
2. *Delete Mutator*: This operator operates on both feasible and infeasible routes. It randomly removes an intermediate node from the selected route.
3. *Insert Internal Waypoint mutator*: This mutator imagines an oblong shape around the two waypoints that form a segment, then inserts a new waypoint within that shape.

4. *Smooth turn operator*: This operator places two new waypoints on the midpoint of the segments either side of the selected segment, then deletes the selected waypoint, the result being that it will create segments with smoother turns that the previous.
5. *Split Segment mutator:* This mutator splits a segment in two, placing a waypoint on a segments mid-point. This is used to help ensure segment lengths are shorter than the maximum segment length.
6. *Approach Mutator*: This operator replaces the penultimate waypoint with a new waypoint randomly placed within the approach cone of an airport.

3.4 Route Feasibility

A route will be feasible if it passes through no excluded airspace, and if it does not make any turns at too great of an angle. If a route is infeasible, then it will not have its fitness evaluated. It will instead have its fitness set equal to the sum of the largest feasible fitness, and the number of infeasible segments in the route.

3.5 Evaluating Fitness

For each Individual Route r_i the routes fitness $F(r_i)$ shall be defined as follows:

$$F(r_i) = \sum\{ l_r * w_l + t_r * w_t + pfo_r * w_{pfo} + fb_r * w_{fb} \} \quad (1)$$

Where each w is the weight of the corresponding value. Each value is outlined below:

1. *l (Total route length): Total length in kilometres of the route.* Where $len(s_{ri})$ is the length of a single segment (s_{ri}) of the route.

$$l_r = \sum_{i=0}^{n}\{len(s_{ri})\} \quad (2)$$

2. *t (Total route time)*: *Total flight time for the route.* Where spd_{sri} *is the speed restriction of the segment.*

$$t_r = \sum_{i=0}^{n}\{spd_{sri} * len(s_{ri})\} \quad (3)$$

3. *pfo (Population overflown in each flight band)*: Where pfo_{r0} represents the population in the height band FL0 – FL40, pfo_{r40} represents the population in the height band FL40 – FL70, pfo_{r70} represents the population in the height band FL70+.

$$pfo_r = \sum pfo_{r0} + pfo_{r40} + pfo_{r70} \quad (4)$$

4. *Total fuel burned*: For a given number of aircraft, flying the generated routes, what is the total amount of fuel burned in KG. Where fbr_r represents the fuel burn rate for the aircraft on the selected route. This can be adapted on a per route basis.

$$\sum_{i=0}^{n}\{fbr_r * len(s_{ri})\} \quad (5)$$

This algorithm measures fitness against one objective *F(r)*. The aim is to minimise *F(r).* The use of the weights *w* allows for the prioritisation of different objectives to meet the users' needs. This means that one can decide whether to prioritise environmental disruption due to noise over CO_2 emissions and cost by adjusting the weight values.

This is inspired by [4] however as it is not in real time, it considers some of the concerns from [6] to introduce novel mutators to insure low complexity when multiple routes are introduced.

4 The ROUTE Algorithm

The ROUTE algorithm is designed using computational evolution. The algorithm will initially generate a population of individual routes, these routes will then be evaluated according to a set of viability criteria. Those individual routes who meet the viability criteria are then evaluated for fitness. This is done for efficiency as fitness calculations are computationally expensive. Once evaluated the routes are then randomly selected using the roulette method for mutation and crossover. This roulette method is a method of random selection where more fit individuals are more likely to be selected however less fit individuals are still able to be selected, this preserves diversity. Once selected the individuals are subject to either mutation or crossover, or are directly passed through to the next generation, the chances of any of these three operators being applied are equal. The routes are then re-evaluated and the process loops and continues until the end condition is reached. The end condition in ROUTE is either a limit on number of generations, or the number of generations progressed without a significant change. This process is outlined in Figure 1 below.

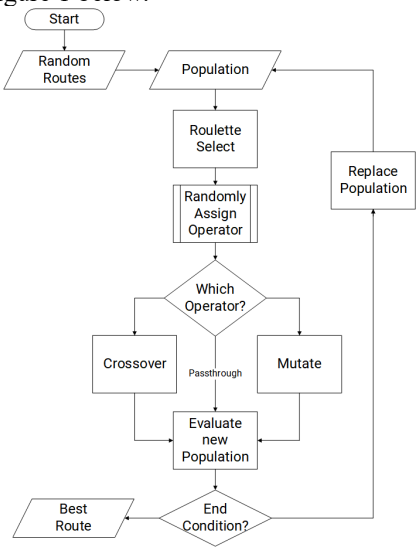

Fig. 1. ROUTE Algorithm flow diagram

In this algorithm, a chromosome represents a group of routes. Each route is a sequence of points, initially randomly selected, between fixed entrance and exit points. All the candidate routes form the population, this is formed of P groups of routes.

For evolution, a set S is selected where $S \leq P$ this set forms the candidates for genetic mutation and crossover. Individuals from P are selected using a roulette wheel of P slots, where the probability of an individual being selected is inversely proportional to the fitness of and individual. During roulette selection, an individual may be selected more than once.

After the selection of S, the set is iterated through, randomly selecting whether to mutate or not each individual, and where selected for mutation, each mutator has an equal chance of being selected. Where a mutator is not applicable to an individual, another mutator will be selected from the pool, with that mutator removed. Once mutated fitness is re-evaluated, and the candidate is added to a new extended population.

Once the set S has been exhausted the new extended population is evaluated and the fittest P individuals are retained, while the lesser fit individuals are discarded.

Once a proposed solution is suitably fit, this solution is selected as the optimal solution. Whether a solution is suitably fit can be chosen after a fixed number of generations or evaluated by the algorithm.

The steps of the algorithm can be summarised as follows:

```
DATA: Population, ViablePool, NonViablePool
START
  randomize Population of size P
  while end condition not met do
    for each individual in population do
      evaluateViabilityScore() of individual
      if individual's viability = 0 then
        evaluate fitness of individual
        add individual to ViablePool
      else if viability score > 0 then
        add individual to NonViablePool
      end if
    end for
    for each individual in NonViablePool do
      individual's score = population's highest fitness +
                                        individual's viability
    end for
    combine ViablePool and NonViablePool to form candidates
    sort candidates by fitness from low to high
    assign candidates descending rank from first to last
    breedNextGeneration() and add children to
                                          population in order
    reduce population to P members
  end while
```

The 'breedNextGeneration' step is summarised in the algorithm below.

```
DATA Candidates, Population, Children
```

```
function breedNextGeneration()
START
  while Candidates < S do
    candidate = selectUsingRouletteWheel(Population)
    add candidate to Candidates
  end while
  for each candidate do
    if randomNumber(0 to 1) < mutationChance do
      mutator = select mutator at randomNumber(1 to 8)
      mutate candidate with mutator and add to children
    end if
  end for
  return children
END
```

The roulette wheel selection method ensures that while you are more likely to select fitter individuals, there is a chance to also select the least fit individuals. This ensures that the population does not stagnate. The below algorithm briefly outlines the simple and efficient method for selecting a candidate using a roulette wheel implementation:

```
function selectUsingRouletteWheel ()
START
  for each candidate do
    totalRank += candidateRank
  end for
  targetRank = totalRank * RandomNumber(0.00 to 1.00)
  for each candidate do
    targetRank -= candidateRank
    if targetRank < 0 do
      return candidate
    end if
  end for
  return last candidate
END
```

Lastly the Viability of an individual is described as the sum of the deviance between the upper and lower acceptable bounds and the actual measurement for each waypoint in a route. This non-binary approach means that a small improvement, even though it may not result in a viable route would be preferred as it is a step closer to a viable route.

For example, the following algorithm describes the calculation of the total viability of the angles between segments in the route.

```
DATA TotalViability
function evaluateViabilityScore()
START
  for each waypoint in route do
    firstLineSegment  = lineSegment(waypoint to waypoint+1)
    secondLineSegment = lineSegment(waypoint+1 to waypoint+2)
```

```
      angle = angleBetween(firstLineSegment, secondLineSegment)
      if angle > allowedAngle then
        TotalViability += angle – allowedAngle
      end if
    end for
    return TotalViability
  END
```

Once evolution is complete, the fittest routes are selected. Each segment of each route is iterated through. Where two routes intersect each other a two sub routes are created from the point of intersection onwards. Each sub route is evaluated for fitness, and the fittest sub route replaces the less fit sub route in both routes. This minimises complexity as where the routes share commonality they will merge. This is illustrated in figure two below.

5 Experimental Results

The ROUTE algorithm was implemented using Java 8 run on a Core i7 PC running Windows 10. In order to validate that ROUTE reaches its objectives, we will present evidence of the effect of changing weight parameters, as well as the effects of changing constraints on the route, lastly presenting the overall effectiveness of the solution.

In the Figures below population is represented in the coloured grid beneath the routes drawn. Purple represents higher population in a square, and pink represents less population. The red line represents the approach, the top of the approach is where the routes are attempting to converge, they would then fly the approach to the runway at the opposite end.

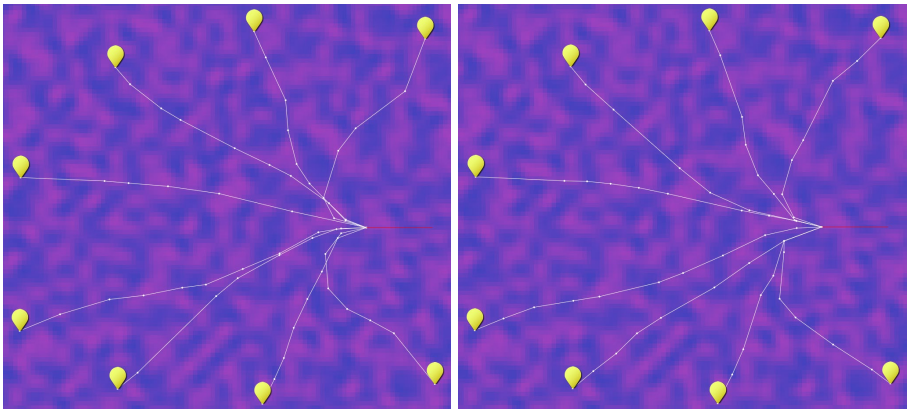

Fig. 2. Completed Routes

Fig. 2 shows how a set of routes can evolve together. The left shows routes evolved, and the right shows how these routes can be simplified to prevent unnecessary overlapping.

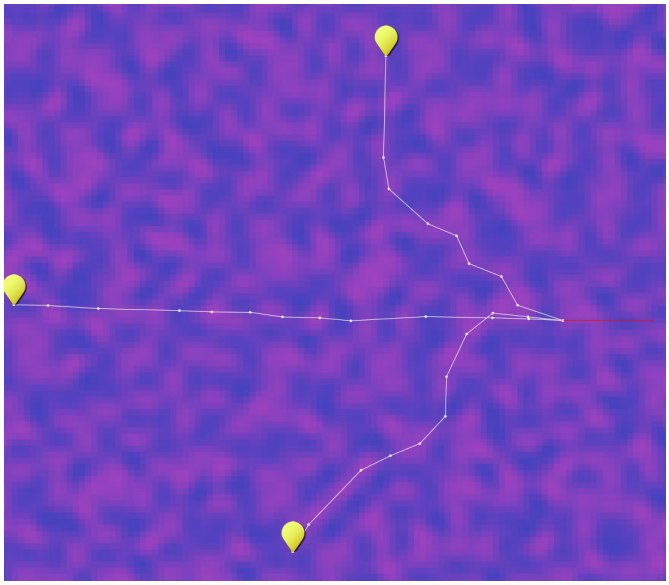

Fig. 3. Showing one route with a length weight of zero.

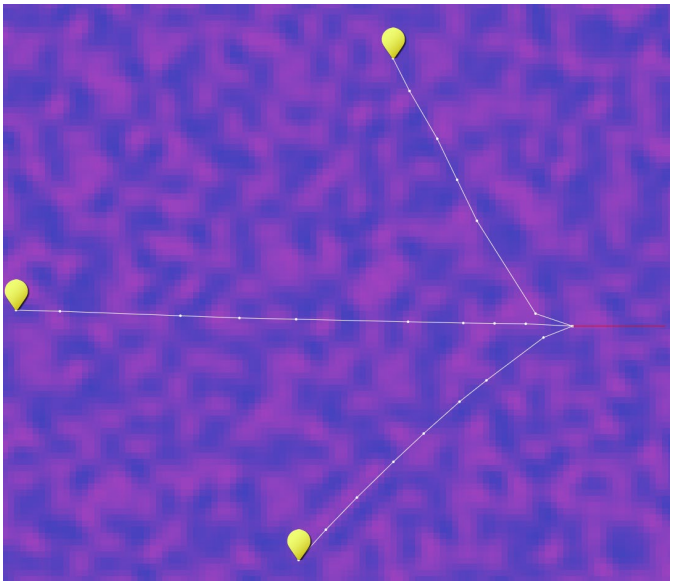

Fig. 4. Showing one route with a length weight of 10. This tends to take a more direct route.

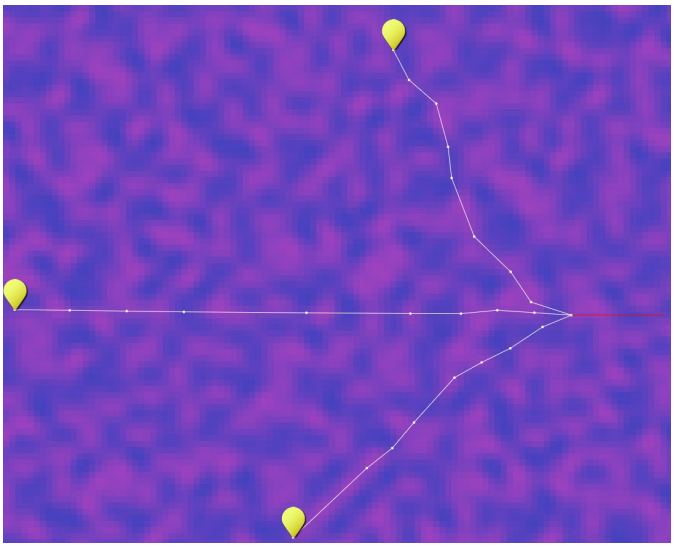

Fig. 5. Balanced length and population

Fig. 3 Shows a closer representation of how a lower priority ($w_l = 0$) results in routes that more carefully avoid population. This is compared with Fig. 4 which shows how a higher priority ($w_l = 10$) can lead to an almost direct route. Lastly Fig. 5 shows how a more appropriate weight ($w_l = 1$) can lead to a route who is a better compromise, of being simpler with fewer route corrections, while also showing some tendency to avoid populated areas.

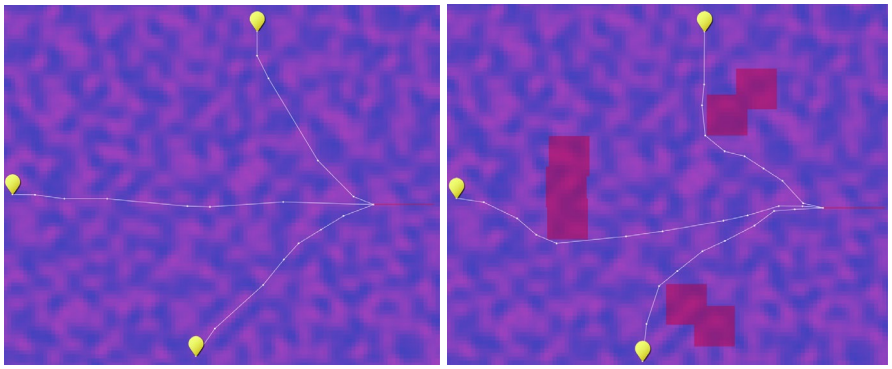

Fig. 6. Demonstrating the avoidance of restricted areas (red boxes)

Fig. 6 shows best how viability in ROUTE can affect a routes shape. Before ROUTE will evaluate the fitness of a route it must first evaluate whether it breaks any restrictions, for example if a turn is to sharp, or if it enters a restricted area. In this example, the route will first need to not enter the red box, before it is then optimised, notice therefore how it will avoid but stick close to the restricted areas. This demonstrates the ability of the ability to conform to viability requirements.

In addition to visually confirming these results, we have included below the results from a series of tests, illustrating the effects of changing the algorithm configuration on the generated routes.

Firstly, we look at the effect of changing weights to determine whether the route generated should be shorter or fly over fewer people.

Table 1. Higher population weight (weight=10).

	N	Minimum	Maximum	Mean	Std. Deviation
Length	5	1056.42	1068.35	1061.3707	4.91244
Population	5	199.20	208.03	203.6226	3.83253

Table 2. Higher length weight (weight=10).

	N	Minimum	Maximum	Mean	Std. Deviation
Length	5	1011.51	1013.87	1012.3594	1.07681
Population	5	257.42	265.11	260.5541	2.95241

As should be expected, we can see that the mean length is smaller when the length weight is higher, similarly for the population it is smaller when the population weight is higher. However, interestingly the standard deviation is higher in the measurement with the lower weight. It could be concluded that due to the lower weight, the value is not as important to the algorithm, therefore it becomes less consistently optimised because the algorithm will it to boost other statistics.

Next, we look at how changing the number of generations can impact the performance of the algorithm.

Table 3. Fewer generations (generations=10).

	N	Minimum	Maximum	Mean	Std. Deviation
Fitness	5	17912.91	18039.44	17971.4361	50.49479

Table 4. Greater generations (generations=100).

	N	Minimum	Maximum	Mean	Std. Deviation
Population	5	17827.42	17927.83	17875.2560	42.76566

Here it is possible to see the two main benefits of increasing the number of generations used in evolution. Firstly, the longer search results in better results, this can be seen as all values of fitness are lower in the test allowed to run for more generations.

Secondly, we also see a smaller standard deviation on the longer running sample. This suggest that the results are groups closer together. From this it can be concluded that with more time a given result is more reliably the `best' route.

Lastly, we consider the effects of changing the population size. This is to see whether the computational expense of more individuals is rewarded with higher quality results.

Table 5. Smaller population size (population=20).

	N	Minimum	Maximum	Mean	Std. Deviation

Fitness	5	17929.01	18079.07	17993.6140	63.21561

Table 6. Larger population size (population=300).

	N	Minimum	Maximum	Mean	Std. Deviation
Population	5	17795.27	17942.86	17882.3460	59.22986

Here it is possible to see that with a larger population the deviation is smaller, this suggests that with more individuals each run results in a closer result. This highlights the drawback of a genetic algorithm, as each run is based off a different random start point, the outcomes are not necessarily repeatable. However, seeing the smaller standard deviation with the higher population size suggests that a way to mitigate this pitfall is using a larger population. This small deviation suggests that the results are more similar, and therefore more consistent.

6 Conclusions

This paper has proposed ROUTE, an evolutionary approach to the design of airspace. The paper has demonstrated ways in which a route can be influenced by shifting priorities, to create a route fit for a given situation. This can be seen in the way the priorities change as a route gets closer to the ground, with a route following more contoured paths when it is lower in height.

We believe this work has identified the viability of generating airspace structures using computational aided methods. However, more work should be done to evaluate what would be necessary to validate such an approach for operational use in such a safety critical industry.

Moreover, improvements should be explored in simplifying airspace designs to promote simplicity. While this research begins to explore this problem, it looks for commonality in the routes, a better option may be to incorporate this as part of the genetic algorithm itself. This limitation of the algorithm, as well as the inability to compare it to operational data will need to be addressed before this approach could see operational use.

The method initially outlined here, however, has great potential to create modern and adaptive navigational infrastructure, and could have applications beyond airspace, and future efforts could consider other areas of route planning which would benefit from a balanced approach to route design.

References

[1] EUROCONTROL, “European Airspace Concept Handbook for PBN Implementation.”

[2] S. Timar, G. Hunter, and J. Post, “Assessing the Benefits of NextGen Performance Based Navigation (PBN),” 2013.

[3] X. Zhang and H. Duan, “An improved constrained differential evolution

algorithm for unmanned aerial vehicle global route planning," *Appl. Soft Comput. J.*, vol. 26, pp. 270–284, Jan. 2015.

[4] C. Zheng, L. Li, F. Xu, F. Sun, and M. Ding, "Evolutionary route planner for unmanned air vehicles," *IEEE Trans. Robot.*, vol. 21, no. 4, pp. 609–620, 2005.

[5] A. Basu, J. S. B. Mitchell, and G. Sabhnani, "Geometric Algorithms for Optimal Airspace Design and Air Traffic Controller Workload Balancing," in *2008 Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, Philadelphia, PA: Society for Industrial and Applied Mathematics, 2008, pp. 75–89.

[6] D. Delahaye and S. Puechmorel, "3D airspace design by Evolutionary computation," in *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, 2008, p. 3.B.6-1-3.B.6-13.

[7] CAA, "Performance-based Navigation Airspace Design Guidance : Noise mitigation considerations when designing PBN departure and arrival procedures."

[8] D. Whitley, "A Genetic Algorithm Tutorial."

[9] M. Xue, "Design Analysis of Corridors-in-the-sky," Santa Cruz, CA, United States, 2008.

[10] M. Xue and P. H. Kopardekar, "High-Capacity Tube Network Design Using the Hough Transform," *J. Guid. Control. Dyn.*, vol. 32, no. 3, pp. 788–795, 2009.

[11] R. Ehrmanntraut and S. McMillan, "Airspace design process for dynamic sectorisation," *AIAA/IEEE Digit. Avion. Syst. Conf. - Proc.*, pp. 1–9, 2007.

[12] K. R. Sprong, B. M. Haltli, J. S. Dearmon, and S. Bradley, "IMPROVING FLIGHT EFFICIENCY THROUGH TERMINAL AREA RNAV."

[13] R. H. Hogenhuis, S. J. Hebly, and H. G. Visser, "Optimization of area navigation noise abatement approach trajectories," *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.*, vol. 225, no. 5, pp. 513–521, 2011.

[14] S. Hartjes, H. G. Visser, and S. J. Hebly, "Optimization of RNAV Noise and Emission Abatement Departure Procedures," no. September, pp. 1–13, 2012.

[15] J. Lovegren and R. J. Hansman, "Estimation of Potential Aircraft Fuel Burn Reduction in Cruise Via Speed and Altitude Optimization Strategies," 2011.

[16] O. Sahin Meric, "Optimum Arrival Routes for Flight Efficiency," *J. Power Energy Eng.*, vol. 3, pp. 449–452, 2015.

[17] NATS, "LAMP Phase 1a airspace change now live - NATS," 2016. [Online]. Available: https://www.nats.aero/news/newsbrief/janfeb-2016/lamp-phase-1a-airspace-change-now-live/. [Accessed: 28-Feb-2019].